# Cortix Documentation

***Release 0.1.0***

**Valmor F. de Almeida, Taha Azzaoui, Gilberto E. Alas**

**Mar 22, 2019**

# CONTENTS

CHAPTER

ONE

# SRC

## 1.1 application

Application class for Cortix.

Cortix: a program for system-level modules coupling, execution, and analysis.

**class** application.**Application**(*app_work_dir*, *config_xml_tree*)

Bases: object

An Application is a singleton class composed of Module objects, and Network objects; the latter involve Module objects in various combinations. Each combination is assigned to a Network object.

**get_module**(*name*)

Returns a module with a given name. None if the name doesn't exist.

**get_network**(*name*)

Returns a network with a given name. None if the name doesn't exist.

**modules**

*list(str)* – List of names of Cortix module objects

**networks**

*list(str)* – List of names of network objects

## 1.2 cortix_driver_template

Cortix driver for guest modules. Module developers must implement the public methods in this driver. Ideally, this implementation should be minimal. Developers should use this class to wrap their module (MyModule) implemented in a file named my_module.py. This file will be placed inside the developer's module directory which is pointed to in the Cortix config.xml file.

**class** cortix_driver_template.**CortixDriverTemplate**(*slot_id*, *input_full_path_file_name*, *exec_full_path_file_name*, *work_dir*, *ports=[]*, *cortix_start_time=0.0*, *cortix_final_time=0.0*, *cortix_time_unit=None*, *cortix_time_step=0.0*)

Bases: object

Cortix driver for guest modules.

**call_ports**(*cortix_time=0.0*)

Call all ports at cortix_time

**execute** (*cortix_time=0.0, timeStep=0.0*)
Evolve system from cortix_time to cortix_time + timeStep

## 1.3 cortix_main

The Cortix class definition.

Cortix: a program for system-level modules coupling, execution, and analysis.

**class** cortix_main.**Cortix** (*name, config_xml_file='cortix-config.xml'*)
Bases: object

The main Cortix class definition. This class encapsulates the concepts of simulations, tasks, and modules, for a given application providing the user with an interface to the simulations.

**run_simulations** (*task_name=None*)
This method runs every simulation defined by the Cortix object. At the moment this is done one simulation at a time.

## 1.4 cortix_module_template

Simple MyModule module template for developers.

**class** cortix_module_template.**MyModule** (*slot_id, input_full_path_file_name, work_dir, ports=[], cortix_start_time=0.0, cortix_final_time=0.0, cortix_time_unit=None*)
Bases: object

MyModule template for Cortix.

**call_ports** (*cortix_time=0.0*)
Developer must implement this method. Transfer data at cortix_time

**execute** (*cortix_time=0.0, cortix_time_step=0.0*)
Developer must implement this method. Evolve system from cortix_time to cortix_time + cortix_time_step

## 1.5 launcher

Launcher functionality of the Cortix Class.

Cortix: a program for system-level modules coupling, execution, and analysis.

**class** launcher.**Launcher** (*importlib_name, mod_lib_parent_dir, module_name, slot_id, input_full_path_file_name, exec_full_path_file_name, work_dir, cortix_param_full_path_file_name, cortix_comm_full_path_file_name, runtime_status_full_path*)
Bases: threading.Thread

The Launcher class handles the main funcitonality of stepping through the simulation time, and monitoring its progress.

**run** ()
Function used to timestep through the modules. Runs the simulation from start to end, and monitors its progress at each time step.

## 1.6 module

Cortix Module class defintion.

Cortix: a program for system-level modules coupling, execution, and analysis.

**class** module.**Module**(*logger*, *importlib_name*, *library_parent_dir*, *config_xml_tree*)

Bases: object

The Module class encapsulates a computational module of some scientific domain.

**diagram**

Return the diagram string from the module manifest or a null place holder.

**execute**(*slot_id*, *runtime_cortix_param_file*, *runtime_cortix_comm_file*)

Spawns a worker process to execute the module.

**get_port_mode**(*port_name*)

Returns the port mode specified by port_name

**get_port_type**(*port_name*)

Returns the port type specified by port_name

**has_port_name**(*port_name*)

Returns true if a port with the name port_name is available in the module.

**importlib_name**

*str* – Module library name

**library_parent_dir**

*str* – Library parent directory

**name**

*str* – Module name

**port_names**

*list(tuple)* – List of names of module's ports

**ports**

*list(tuple)* – Module's ports

## 1.7 network

Network class for the Cortix project. A network defines the connectivity between Cortix modules.

Cortix: a program for system-level modules coupling, execution, and analysis.

**class** network.**Network**(*net_config_node*)

Bases: object

Cortix network class definition. Network class members:

**__config_node: XMLTree** Configuration data in the form of an XML tree.

**__name:str** Name of the network.

**__connectivity: list(dict)** List of dictionaries of connectivity. Dictionary: {'fromModuleSlot': module_slot_name, 'fromPort': use_port_name, 'toModuleSlot': module_slot_name, 'toPort': provide_port_name}.

**__module_slot_names: list(str)** List of names of module slots.

**__runtime_cortix_comm_file_name: dict** Full path filename of the communication file for each module slot. {module_slot_name:full_path_comm_file_name, .:., .:., . . . }. This is initially filled with a null filename at construction time. Later, Simulation.__setup_task() will fill in the information.

__nx_graph:

**__repr__**()
Network to string conversion

**__str__**()
Network to string conversion

**connectivity**
*list(dict)* – List of the network connectivity

**get_runtime_cortix_comm_file_name**(*module_slot_name*)
Returns the cortix comm file that corresponds to module_slot_name. None if otherwise.

**module_slot_names**
*list(str)* – List of network slot names

**name**
*str* – Network name

**nx_graph**
*networkx.MultiDiGraph* – NXGraph corresponding to network

**set_runtime_cortix_comm_file_name**(*module_slot_name*, *full_path_file_name*)
Sets the runtime cortix communications file to the one specified by full_path_file_name

## 1.8 simulation

Simulation class of Cortix.

Cortix: a program for system-level modules coupling, execution, and analysis.

**class** simulation.**Simulation**(*parent_work_dir*, *config_xml_tree*)
Bases: object

Cortix Simulation element as defined in the Cortix config.

**execute**(*task_name=None*)
This method allows for the execution of a simulation by executing each task, if any. Execution proceeds one task at a time.

## 1.9 task

The Cortix Task class definition.

Cortix: a program for system-level modules coupling, execution, and analysis.

**class** task.**Task**(*parent_work_dir=None*, *task_config_node=<cortix.src.utils.xmltree.XMLTree object>*)
Bases: object

A Task is work done by a Simulation handled by Cortix. A Task will use a given Application.

**evolve_time**
*float* – Task final time

**evolve_time_unit**
*str* – Task final time unit

**execute** (*application*)
This method is used to execute (accomplish) the given task.

**name**
*str* – Task name

**runtime_cortix_param_file**
*str* – Task's config file

**set_runtime_cortix_param_file** (*full_path*)
Sets the task config file to the specified file.

**start_time**
*float* – Task initial time

**start_time_unit**
*str* – Task initial time unit

**time_step**
*float* – Magnitude of incremental step in the task's time

**time_step_unit**
*str* – Time step unit

**work_dir**
*str* – Working directory of task

## 1.10 utils

### 1.10.1 set_logger_level

This file contains a helper function used by functions across the Cortix project to set the level of the logger.

set_logger_level.**set_logger_level** (*handler*, *handler_name*, *handler_level*)
This is a helper function that takes in a file/console handler and sets its logger level accordingly.

### 1.10.2 xmltree

This file contains the class definition of *XMLTree*, which aids in parsing the XML configuration files used within the Cortix project.

#### Background

An XML document consists of a collection of elements (nested or not). These elements are containers of information and are the most important components of an XML document. An element container is defined by a start tag and an end tag. A tag starts with an opening angle bracket and finishes with a closing angle bracket as follows

- **element**: <ele_name> </ele_name>, *e.g*, <net_x> </net_x>.

Here *<net_x>* is the start tag and *</net_x>* is the end tag for the element named *net_x*. An element with the same name can be used repeatedly in an XML document. In *Cortix* we will use element names following the Python variable name snake convention. It is also common to call the element name as the tag name. The start tag of an element may have any number of atributes:

- **attributes**: name='value', *e.g.*, <net_x mod_slot='wind:0'> </net_x>

these are name-value pairs; we will use the same name convention for element names when creating attribute names. There may be many attributes in a start tag. Finally everything in between tags of an element, is considered as the element content:

- **content**: <tag> content </tag>, *e.g.*, <net_x mod_slot='wind:0'>189.67 MeV</net_x>

Cortix: a program for system-level modules coupling, execution, and analysis.

**class** `xmltree.`**`XMLTree`**(*xml_tree_node=None*, *xml_tree_file=None*)

Bases: `object`

This class is a wrapper around the XML parser ElementTree and Element. See import statement above. This XML parser is fast but the interface is not very user friendly; hence the motivation for this wrapper. The interface is designed to facilitate the use of XML data within Cortix and its modules. This class generates objects that hold an XML tree: ElementTree. Configuration of Cortix and some runtime files are the primary usage of XMLTree. The construction of an XMLTree object either uses a file with an XML content or an XML branch of an XML tree. This makes it useful throughout Cortix to inspect branches of a configuration XML tree to retrieve data. A node in a tree is the root of a branch. That is, the same thing as an XML element and all its direct sub-elements; described in the Background above.

**`get_all_sub_nodes`**(*tag*)

Returns a list of all direct children nodes in the element tree with tag name.

**Parameters** **`tag`** (*str*) – XML element name or tag name, *e.g.*: *<task></task>*

**Returns** **sub_nodes** – List of XMLTree items.

**Return type** list(*XMLTree*)

**`get_node_attribute`**(*attribute_name*)

Returns the value of the attribute associated with the root node of the element tree, *e.g.* <module type='native'></module>. Attribute name is *type*, value is 'native'.

**Parameters** **`attribute_name`** (*`str`*) –

**Returns** **attribute_value**

**Return type** str

**`get_node_children`**()

Returns a list of the direct sub-elements in the given element (node) containing: the subnode, the tag (element) name, the attributes a list of tuples, and the content (text) of the node. This is not recursive. Recursion can be done by calling this method on the children nodes (that is the first element of the tuple).

**Parameters** **`empty`** –

**Returns** **children** – Tuple: (node, tag, [(attribute name,attribute value),(.,.)…], content). Attribute name and value are string type.

**Return type** list(tuple)

**`get_node_content`**()

Returns the content or text of the XML element.This is what is in between the start and end tags of the element.

**Parameters** **`Empty`** –

**Returns** **content**

**Return type** str

**get_node_tag**()

Returns the tag name associated with the root node of the element tree. This is the element name or tag name, *e.g.*, *<elem_name> </elem_name>*.

**Parameters** **empty** –

**Returns** **tag_name**

**Return type** str

**get_root_node**()

Returns the Element tree's root node.

**Parameters** **empty** –

**Returns** **self.__xml_tree_node**

**Return type** Element

**get_sub_node**(*tag*)

Returns the first subnode (branch) of the element tree specified by the parameter tag.

**Parameters** **tag** (*str*) – This is the XML element name (or tag name).

**Returns** **node** – An XML branch tree starting from *node*.

**Return type** *XMLTree*

CHAPTER

TWO

# MODULIB

## 2.1 pyplot

### 2.1.1 cortix_driver

Cortix driver for the PyPlot module.

**class** cortix_driver.**CortixDriver**(*slot_id*, *input_full_path_file_name*, *exec_full_path_file_name*, *work_dir*, *ports=[]*, *cortix_start_time=0.0*, *cortix_final_time=0.0*, *cortix_time_step=0.0*, *cortix_time_unit=None*)

Bases: object

**call_ports**(*cortix_time=0.0*)
: Call all ports at cortix_time

**execute**(*cortix_time=0.0*, *time_step=0.0*)
: Evolve system from cortix_time to cortix_time + time_step

### 2.1.2 pyplot

PyPlot module.

Author: Valmor F. de Almeida dealmeidav@ornl.gov; vfda Tue Jun 24 01:03:45 EDT 2014

**class** pyplot.**PyPlot**(*slot_id*, *input_full_path_file_name*, *work_dir*, *ports=[]*, *cortix_start_time=0.0*, *cortix_final_time=0.0*, *cortix_time_step=0.0*, *cortix_time_unit=None*)

Bases: object

**call_ports**(*cortix_time=0.0*)
: Transfer data at cortix_time

**execute**(*cortix_time=0.0*, *time_step=0.0*)

### 2.1.3 time_sequence

Valmor F. de Almeida dealmeidav@ornl.gov; vfda

Pyplot module.

This class manages time-sequence data in XML or tabular formats. It is a helper for reading and manipulating stored file data in Cortix. The XML data is a ElementTree object.

Sat Jul 19 12:13:05 EDT 2014

**class** time_sequence.**TimeSequence**(*fileName*, *fileType*, *initialTime=0.0*, *finalTime=0.0*, *time_unit=None*, *logger=None*)

Bases: `object`

**GetNVariables**()

**GetTimeUnit**()

**GetVariableNames**()

**GetVariables**()

**get_name**()

CHAPTER

THREE

# EXAMPLES

## 3.1 console_run

### 3.1.1 droplet_run

Cortix: a program for system-level modules coupling, execution, and analysis.

`droplet_run.run()`

Run the Cortix Droplet example. If Cortix and its dependencies are installed, this program should be executed at the command prompt inside the directory this program resides, namely, cortix/cortix/example/console_run/ directory. At the end of execution a directory with all logging and outputs is left in the work directory, as specified in the cortix-config-droplet.xml file. In this case, /tmp/cortix-droplet-wrk/.

### 3.1.2 main_executor

Cortix: a program for system-level modules coupling, execution, and analysis.

Cortix is a library and it is used by means of a driver. This file is a simple example of a driver. Many Cortix objects can be ran simultaneously; a single object may be sufficient since many simulation/tasks can be ran via one object.

As Cortix evolves additional complexity may be added to this driver and/or other driver examples can be created.

`main_executor.main()`

### 3.1.3 main_mpi

Cortix: a program for system-level modules coupling, execution, and analysis.

Cortix is a library and it is used by means of a driver. This file is a simple example of a driver. Many Cortix objects can be ran simultaneously; a single object may be sufficient since many simulation/tasks can be ran via one object.

As Cortix evolves additional complexity may be added to this driver and/or other driver examples can be created.

`main_mpi.main()`

### 3.1.4 main_pthread

Cortix: a program for system-level modules coupling, execution, and analysis.

Cortix is a library and it is used by means of a driver. This file is a simple example of a driver. Many Cortix objects can be ran simultaneously; a single object may be sufficient since many simulation/tasks can be ran via one object.

As Cortix evolves additional complexity may be added to this driver and/or other driver examples can be created.

```
main_pthread.main()
```

## 3.2 input

### 3.2.1 cortix-config

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# This file is part of the Cortix toolkit environment
# https://cortix.org
#
# All rights reserved, see COPYRIGHT for full restrictions.
# https://github.com/dpploy/cortix/blob/master/COPYRIGHT.txt
#
# Licensed under the University of Massachusetts Lowell LICENSE:
# https://github.com/dpploy/cortix/blob/master/LICENSE.txt
-->

<!-- Configuration of Cortix -->

<cortix_config version="0.1">

<!-- ***************************************************************************** --
↪>
<!--                    CORTIX object starts here                                  --
↪>

<!-- Cortix instantiation definition -->
<!-- NB: XML elements in this level must be unique across other configuration files if
         multiple Cortix objects are used in the cortix-main.py program.
         If this rule is not followed, there will be collision with file in/outputs;
         including results files and logging files.-->

 <name>cortix-dev1</name>

 <logger level="DEBUG">
  <file_handler level="DEBUG"> </file_handler>
  <console_handler level="INFO"> </console_handler>
 </logger>

 <work_dir>/tmp/</work_dir>

 <!-- ============================================================================ --
↪>
 <!--                Simulation object starts here                                 --
↪>

 <!-- Simulation definition -->
 <!-- NB: each simulation has only one application -->

 <simulation name="dev1">

  <logger level="DEBUG">
```

(continues on next page)

(continued from previous page)

```
  <file_handler level="DEBUG"> </file_handler>
  <console_handler level="INFO"> </console_handler>
 </logger>

 <!-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ --
↪>
 <!--                   Tasks objects start here                                  --
↪>

 <!-- Simulation: tasks -->
 <!--NB: each task combines parameters for one named application connectivity-->
 <!--NB: each task name must match a network name below-->

 <task name="solo-pyplot" >
  <start_time unit="hour">1.0</start_time>
  <evolve_time unit="hour">16.0</evolve_time>
  <time_step unit="minute">1.0</time_step>
  <logger level="DEBUG">
   <file_handler level="DEBUG"> </file_handler>
   <console_handler level="INFO"> </console_handler>
  </logger>
 </task>

 <task name="solo-fueldepot" >
  <evolve_time unit="hour">2.0</evolve_time>
  <time_step unit="minute">1.0</time_step>
  <logger level="DEBUG">
   <file_handler level="DEBUG"> </file_handler>
   <console_handler level="INFO"> </console_handler>
  </logger>
 </task>

 <task name="solo-shear" >
  <evolve_time unit="hour">12.0</evolve_time>
  <time_step unit="minute">1.0</time_step>
  <logger level="DEBUG">
   <file_handler level="DEBUG"> </file_handler>
   <console_handler level="INFO"> </console_handler>
  </logger>
 </task>

 <task name="solo-fuel-accum" >
  <evolve_time unit="hour">24.0</evolve_time>
  <time_step unit="minute">1.0</time_step>
  <logger level="DEBUG">
   <file_handler level="DEBUG"> </file_handler>
   <console_handler level="INFO"> </console_handler>
  </logger>
 </task>

 <task name="solo-dissolverA">
  <evolve_time unit="hour">1.0</evolve_time>
  <time_step unit="minute">1.0</time_step>
  <logger level="DEBUG">
   <file_handler level="DEBUG"> </file_handler>
   <console_handler level="INFO"> </console_handler>
  </logger>
```

(continues on next page)

(continued from previous page)

```
  </task>

  <task name="solo-condenser">
   <evolve_time unit="hour">18.0</evolve_time> <!-- 18h max -->
   <time_step unit="minute">1.0</time_step>
   <logger level="DEBUG">
    <file_handler level="DEBUG"> </file_handler>
    <console_handler level="INFO"> </console_handler>
   </logger>
  </task>

  <task name="solo-tank">
   <evolve_time unit="hour">42.0</evolve_time>
   <time_step unit="minute">1.0</time_step>
   <logger level="DEBUG">
    <file_handler level="DEBUG"> </file_handler>
    <console_handler level="INFO"> </console_handler>
   </logger>
  </task>

  <task name="solo-plume" >
   <evolve_time unit="hour">12.0</evolve_time>
   <time_step unit="minute">1.0</time_step>
   <logger level="DEBUG">
    <file_handler level="DEBUG"> </file_handler>
    <console_handler level="INFO"> </console_handler>
   </logger>
  </task>

  <task name="solo-cooltower" >
   <evolve_time unit="hour">1.0</evolve_time>
   <time_step unit="minute">1.0</time_step>
   <logger level="DEBUG">
    <file_handler level="DEBUG"> </file_handler>
    <console_handler level="INFO"> </console_handler>
   </logger>
  </task>

  <task name="solo-solventxtract" >
   <evolve_time unit="hour">14.0</evolve_time>
   <time_step unit="minute">1.0</time_step>
   <logger level="DEBUG">
    <file_handler level="DEBUG"> </file_handler>
    <console_handler level="INFO"> </console_handler>
   </logger>
  </task>

  <task name="fueldepot-chopper">
<!--   <evolve_time unit="hour">108.0</evolve_time> --> <!-- all assemblies -->
<!--   <evolve_time unit="hour">36.0</evolve_time> --> <!-- 1 assembly -->
   <evolve_time unit="hour">108.0</evolve_time>
   <time_step unit="minute">1.0</time_step>
   <logger level="DEBUG">
    <file_handler level="DEBUG"> </file_handler>
    <console_handler level="INFO"> </console_handler>
   </logger>
  </task>
```

(continues on next page)

(continued from previous page)

```
  <task name="fueldepot-chopper-storage">
<!--   <evolve_time unit="hour">24.0</evolve_time> -->
   <evolve_time unit="hour">12.0</evolve_time>
   <time_step unit="minute">1.0</time_step>
   <logger level="DEBUG">
    <file_handler level="DEBUG"> </file_handler>
    <console_handler level="INFO"> </console_handler>
   </logger>
  </task>

  <task name="fueldepot-dissolverFpot">
<!--   <evolve_time unit="hour">28.0</evolve_time> -->
<!--   <evolve_time unit="hour">42.0</evolve_time>  -->
<!--   <evolve_time unit="hour">18.0</evolve_time>  -->
   <evolve_time unit="hour">35.0</evolve_time>
   <time_step unit="minute">1.0</time_step>
   <logger level="DEBUG">
    <file_handler level="DEBUG"> </file_handler>
    <console_handler level="INFO"> </console_handler>
   </logger>
  </task>

  <task name="fueldepot-chopper-dissolver">
<!--   <evolve_time unit="hour">28.0</evolve_time> -->
<!--   <evolve_time unit="hour">42.0</evolve_time>  -->
<!--   <evolve_time unit="hour">18.0</evolve_time>  -->
   <evolve_time unit="hour">18.0</evolve_time>
   <time_step unit="minute">1.0</time_step>
   <logger level="DEBUG">
    <file_handler level="DEBUG"> </file_handler>
    <console_handler level="INFO"> </console_handler>
   </logger>
  </task>

  <task name="fueldepot-chopper-dissolver-tank">
<!--   <evolve_time unit="hour">24.0</evolve_time> -->
<!--   <evolve_time unit="hour">17.0</evolve_time>  one batch -->
   <evolve_time unit="hour">42.0</evolve_time>
   <time_step unit="minute">1.0</time_step>
   <logger level="DEBUG">
    <file_handler level="DEBUG"> </file_handler>
    <console_handler level="INFO"> </console_handler>
   </logger>
  </task>

  <task name="tank-feedprep">
   <evolve_time unit="hour">42.0</evolve_time>
   <time_step unit="minute">1.0</time_step>
   <logger level="DEBUG">
    <file_handler level="DEBUG"> </file_handler>
    <console_handler level="INFO"> </console_handler>
   </logger>
  </task>

  <task name="fueldepot-chopper-dissolver-tank-feedprep">
<!--   <evolve_time unit="hour">28.0</evolve_time> -->
```

(continues on next page)

(continued from previous page)

```
<!--   <evolve_time unit="hour">16.0</evolve_time> -->
   <evolve_time unit="hour">42.0</evolve_time>
   <time_step unit="minute">1.0</time_step>
   <logger level="DEBUG">
    <file_handler level="DEBUG"> </file_handler>
    <console_handler level="INFO"> </console_handler>
   </logger>
  </task>

  <task name="fueldepot-chopper-dissolver-tank-feedprep-solventxtract">
<!--   <evolve_time unit="hour">28.0</evolve_time> -->
   <evolve_time unit="hour">14.0</evolve_time>
   <time_step unit="minute">1.0</time_step>
   <logger level="DEBUG">
    <file_handler level="DEBUG"> </file_handler>
    <console_handler level="INFO"> </console_handler>
   </logger>
  </task>

  <task name="shear-dissolve-offgas">
   <evolve_time unit="hour">22.0</evolve_time>
<!-- <evolve_time unit="hour">5.0</evolve_time> -->
   <time_step unit="minute">1.0</time_step>
   <logger level="DEBUG">
    <file_handler level="DEBUG"> </file_handler>
    <console_handler level="INFO"> </console_handler>
   </logger>
  </task>

  <task name="shear-dissolve">
   <evolve_time unit="hour">24.0</evolve_time>
   <time_step unit="minute">1.0</time_step>
   <logger level="DEBUG">
    <file_handler level="DEBUG"> </file_handler>
    <console_handler level="INFO"> </console_handler>
   </logger>
  </task>

  <task name="shear-double-dissolve-single-condense">
   <evolve_time unit="hour">14.0</evolve_time>
   <time_step unit="minute">1.0</time_step>
   <logger level="DEBUG">
    <file_handler level="DEBUG"> </file_handler>
    <console_handler level="INFO"> </console_handler>
   </logger>
  </task>

  <!--                  Tasks objects end here                                --
↪>
  <!-- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ --
↪>

  <!-- ########################################################################### --
↪>
  <!--                  Application object starts here                        --
↪>
```

(continues on next page)

(continued from previous page)

```
  <!-- Simulation: application -->
  <!-- NB: each simulation has only one application -->

  <application name="dev1">

   <module_library name='headend-lib'>
    <parent_dir>
      /home/dealmeida/mac-dealmeida/gentoo-home/work/codes/reprocessing
    </parent_dir>
   </module_library>

   <logger level="DEBUG">
    <file_handler level="DEBUG"> </file_handler>
    <console_handler level="INFO"> </console_handler>
   </logger>

   <!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --
↪>
   <!--                  Modules objects starts here                                 --
↪>
   <!-- Application: modules set -->

   <!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --
↪>
   <module name="fueldepot" type="native">
    <library name='headend-lib'>
     <parent_dir>
      /home/dealmeida/mac-dealmeida/gentoo-home/work/codes/reprocessing
     </parent_dir>
    </library>
    <input_file_name>fueldepot.input</input_file_name>
    <input_file_path>/home/dealmeida/mac-dealmeida/gentoo-home/work/codes/
↪reprocessing/cortix-dev/input/</input_file_path>
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Ports have four types: 1) use, 2) provide, 3) input 4) output -->
    <!-- The *input*, and *output* types allow for module self-connection; see␣
↪network section  -->
    <port type="input"   mode="directory" multiplicity="1">westinghouse-14x14</port>
    <port type="input"   mode="directory" multiplicity="1">mark-31a</port>
    <port type="use"     mode="directory" multiplicity="1">fuel-input</port>
    <port type="use"     mode="file.xml"  multiplicity="1">go-signal</port>
    <port type="provide" mode="file.shlv" multiplicity="1">fuel-bundle</port>
    <port type="provide" mode="file.shlv" multiplicity="1">fuel-bucket</port>
    <port type="output"  mode="file.any"  multiplicity="1">persistent-output</port>
   </module>

   <!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --
↪>
   <module name="chopper" type="native">
    <library name='headend-lib'>
     <parent_dir>
      /home/dealmeida/mac-dealmeida/gentoo-home/work/codes/reprocessing
     </parent_dir>
    </library>
```

(continues on next page)

(continued from previous page)

```
    <input_file_name>chopper.input</input_file_name>
    <input_file_path>/home/dealmeida/mac-dealmeida/gentoo-home/work/codes/
↪reprocessing/cortix-dev/input/</input_file_path>
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Ports have four types: 1) use, 2) provide, 3) input 4) output -->
    <!-- The *input*, and *output* types allow for module self-connection; see␣
↪network section  -->
    <port type="use"     mode="file.shlv" multiplicity="1">fuel-bundle</port>
    <port type="use"     mode="file.xml"  multiplicity="1">go-signal</port>
    <port type="provide" mode="file.shlv" multiplicity="1">fuel-segments</port>
    <port type="provide" mode="file.xml"  multiplicity="1">state</port>
    <port type="provide" mode="file.xml"  multiplicity="1">status-signal</port>
    <port type="output"  mode="file.any"  multiplicity="1">persistent-output</port>
   </module>

   <!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --
↪>
   <module name="oldchopper" type="native">
    <library name='vfda-lib'>
     <parent_dir>
      /home/dealmeida/mac-dealmeida/gentoo-home/work/codes/reprocessing
     </parent_dir>
    </library>
    <input_file_name>oldchopper.input</input_file_name>
    <input_file_path>/home/dealmeida/mac-dealmeida/gentoo-home/work/codes/
↪reprocessing/cortix-dev/input/</input_file_path>
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Ports have four types: 1) use, 2) provide, 3) input 4) output -->
    <!-- The *input*, and *output* types allow for module self-connection; see␣
↪network section  -->
    <port type="input"     mode="file.xml" multiplicity="1">Fuel_Solid</port>
    <port type="input"     mode="file.xml" multiplicity="1">Gas_Release</port>
    <port type="input"     mode="file.xml" multiplicity="1">Particulate</port>
    <port type="use"       mode="file.xml" multiplicity="1">solids-input</port>
    <port type="use"       mode="file.xml" multiplicity="1">gas-input</port>
    <port type="use"       mode="file.xml" multiplicity="1">fines-input</port>
    <port type="provide"   mode="file.xml" multiplicity="1">solids</port>
    <port type="provide"   mode="file.xml" multiplicity="1">off-gas</port>
    <port type="provide"   mode="file.xml" multiplicity="1">fines</port>
   </module>

   <!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --
↪>
   <module name="storage" type="native">
    <input_file_name>storage.input</input_file_name>
    <input_file_path>/home/dealmeida/mac-dealmeida/gentoo-home/work/codes/
↪reprocessing/cortix-dev/input/</input_file_path>
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
```

(continues on next page)

(continued from previous page)

```
    <!-- Ports have four types: 1) use, 2) provide, 3) input 4) output -->
    <!-- The *input*, and *output* types allow for module self-connection; see␣
↪network section  -->
    <port type="input"   mode="file.xml" multiplicity="1">chopper-data-test</port>
    <port type="input"   mode="file.xml" multiplicity="1">dissolver-data-test</port>
    <port type="use"     mode="file.xml" multiplicity="1">fuel-segments</port>
    <port type="use"     mode="file.xml" multiplicity="1">withdrawal-request</port>
    <port type="provide" mode="file.xml" multiplicity="1">status-signal</port>
    <port type="provide" mode="file.xml" multiplicity="1">fuel-segments-on-demand</
↪port>
    <port type="provide" mode="file.xml" multiplicity="1">off-gas</port>
    <port type="provide" mode="file.xml" multiplicity="1">state</port>
   </module>

   <!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --
↪>
   <module name="dissolverA" type="native">
    <library name='vfda-lib'>
     <parent_dir>
      /home/dealmeida/mac-dealmeida/gentoo-home/work/codes/reprocessing
     </parent_dir>
    </library>
    <input_file_name>dissolver.input</input_file_name>
    <input_file_path>/home/dealmeida/mac-dealmeida/gentoo-home/work/codes/
↪reprocessing/cortix-dev/input/</input_file_path>
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Ports have four types: 1) use, 2) provide, 3) input 4) output -->
    <!-- The *input*, and *output* types allow for module self-connection; see␣
↪network section  -->
    <port type="input"   mode="file.xml" multiplicity="1">dissolver-solo-input</port>
    <port type="use"     mode="file.xml" multiplicity="1">solids-input</port>
    <port type="use"     mode="file.xml" multiplicity="1">solids</port>
    <port type="use"     mode="file.xml" multiplicity="1">condensate</port>
    <port type="provide" mode="file.xml" multiplicity="1">signal</port>
    <port type="provide" mode="file.xml" multiplicity="1">state</port>
    <port type="provide" mode="file.xml" multiplicity="1">vapor</port>
    <port type="provide" mode="file.xml" multiplicity="1">product</port>
   </module>

   <!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --
↪>
<!--      /home/dealmeida/mac-dealmeida/gentoo-home/work/codes/reprocessing  -->
   <module name="dissolver" type="native">
    <library name='headend-lib'>
     <parent_dir>
      /home/dealmeida/mac-dealmeida/gentoo-home/work/codes/reprocessing
     </parent_dir>
    </library>
    <input_file_name>dissolver.input</input_file_name>
    <input_file_path>/home/dealmeida/mac-dealmeida/gentoo-home/work/codes/
↪reprocessing/cortix-dev/input/</input_file_path>
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
```

(continues on next page)

(continued from previous page)

```
    </logger>
    <!-- Ports have four types: 1) use, 2) provide, 3) input 4) output -->
    <!-- The *input*, and *output* types allow for module self-connection; see␣
↪network section  -->
    <port type="input"   mode="file.xml"  multiplicity="1">dissolver-solo-input</port>
    <port type="use"     mode="file.xml"  multiplicity="1">solids-input</port>
    <port type="use"     mode="file.xml"  multiplicity="1">go-signal</port>
    <port type="use"     mode="file.xml"  multiplicity="1">fuel-segments</port>
    <port type="use"     mode="file.xml"  multiplicity="1">condensate</port>
    <port type="provide" mode="file.xml"  multiplicity="1">status-signal</port>
    <port type="provide" mode="file.xml"  multiplicity="1">state</port>
    <port type="provide" mode="file.shlv" multiplicity="1">product</port>
    <port type="provide" mode="file.xml"  multiplicity="1">product-plot</port>
    <port type="provide" mode="file.shlv" multiplicity="1">vapor</port>
    <port type="provide" mode="file.dat"  multiplicity="1">vapor-table</port>
    <port type="output"  mode="file.any"  multiplicity="1">persistent-output</port>
   </module>

   <!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --
↪>
   <module name="tank" type="native">
    <library name='ornl-lib'>
     <parent_dir>
      /home/dealmeida/mac-dealmeida/gentoo-home/work/codes/reprocessing
     </parent_dir>
    </library>
    <input_file_name>tank.input</input_file_name>
    <input_file_path>/home/dealmeida/mac-dealmeida/gentoo-home/work/codes/
↪reprocessing/cortix-dev/input/</input_file_path>
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Ports have four types: 1) use, 2) provide, 3) input 4) output -->
    <!-- The *input*, and *output* types allow for module self-connection; see␣
↪network section  -->
    <port type="input"   mode="file.shlv" multiplicity="1">tank-solo-input</port>
    <port type="use"     mode="file.shlv" multiplicity="1">liquid</port>
    <port type="use"     mode="file.xml"  multiplicity="1">withdrawal-request</port>
    <port type="use"     mode="file.xml"  multiplicity="1">go-signal</port>
    <port type="provide" mode="file.xml"  multiplicity="1">status-signal</port>
    <port type="provide" mode="file.xml"  multiplicity="1">liquid-on-demand</port>
    <port type="provide" mode="file.shlv" multiplicity="1">product</port>
    <port type="provide" mode="file.xml"  multiplicity="1">off-gas</port>
    <port type="provide" mode="file.xml"  multiplicity="1">state</port>
    <port type="output"  mode="file.any"  multiplicity="1">persistent-output</port>
   </module>

   <!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --
↪>
   <module name="feedprep" type="native">
    <input_file_name>feedprep.input</input_file_name>
    <input_file_path>/home/dealmeida/mac-dealmeida/gentoo-home/work/codes/
↪reprocessing/cortix-dev/input/</input_file_path>
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
```

(continues on next page)

(continued from previous page)

```
    </logger>
    <!-- Ports have four types: 1) use, 2) provide, 3) input 4) output -->
    <!-- The *input*, and *output* types allow for module self-connection; see␣
↪network section  -->
    <port type="use"     mode="file.shlv" multiplicity="1">liquid</port>
    <port type="use"     mode="file.xml"  multiplicity="1">withdrawal-request</port>
    <port type="use"     mode="file.xml"  multiplicity="1">go-signal</port>
    <port type="provide" mode="file.xml"  multiplicity="1">status-signal</port>
    <port type="provide" mode="file.xml"  multiplicity="1">liquid-on-demand</port>
    <port type="provide" mode="file.shlv" multiplicity="1">product</port>
    <port type="provide" mode="file.xml"  multiplicity="1">product-plot</port>
    <port type="provide" mode="file.xml"  multiplicity="1">off-gas</port>
    <port type="provide" mode="file.xml"  multiplicity="1">state</port>
    <port type="output"  mode="file.any"  multiplicity="1">persistent-output</port>
   </module>

   <!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --
↪>
   <module name="solventxtract" type="native">
    <input_file_name>solventxtract.input</input_file_name>
    <input_file_path>/home/dealmeida/mac-dealmeida/gentoo-home/work/codes/
↪reprocessing/cortix-dev/input/</input_file_path>
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Ports have four types: 1) use, 2) provide, 3) input 4) output -->
    <!-- The *input*, and *output* types allow for module self-connection; see␣
↪network section  -->
    <port type="input"   mode="file.shlv" multiplicity="1">solventxtract-solo-input</
↪port>
    <port type="use"     mode="file.shlv" multiplicity="1">liquid-input</port>
    <port type="use"     mode="file.shlv" multiplicity="1">liquid</port>
    <port type="use"     mode="file.xml"  multiplicity="1">go-signal</port>
    <port type="provide" mode="file.xml"  multiplicity="1">status-signal</port>
    <port type="provide" mode="file.shlv" multiplicity="1">aqueous</port>
    <port type="provide" mode="file.shlv" multiplicity="1">organic</port>
    <port type="provide" mode="file.xml"  multiplicity="1">off-gas</port>
    <port type="provide" mode="file.xml"  multiplicity="1">state</port>
    <port type="provide" mode="file.xml"  multiplicity="1">product-plot</port>
   </module>

   <!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --
↪>
   <module name="condenser" type="native">
    <library name='headend-lib'>
     <parent_dir>
      /home/dealmeida/mac-dealmeida/gentoo-home/work/codes/reprocessing
     </parent_dir>
    </library>
    <input_file_name>condenser.input</input_file_name>
    <input_file_path>/home/dealmeida/mac-dealmeida/gentoo-home/work/codes/
↪reprocessing/cortix-dev/input/</input_file_path>
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
```

(continues on next page)

(continued from previous page)

```
    <!-- Ports have four types: 1) use, 2) provide, 3) input 4) output -->
    <!-- The *input*, and *output* types allow for module self-connection; see␣
↪network section  -->
    <port type="input"   mode="file.shlv" multiplicity="1">condenser-solo-input</port>
    <port type="use"     mode="file.shlv" multiplicity="1">vapor</port>
    <port type="provide" mode="file.shlv" multiplicity="1">off-gas</port>
    <port type="provide" mode="file.shlv" multiplicity="1">condensate</port>
    <port type="provide" mode="file.xml"  multiplicity="1">hot-water</port>
    <port type="provide" mode="file.xml"  multiplicity="1">state</port>
    <port type="output"  mode="file.any"  multiplicity="1">persistent-output</port>
   </module>

   <!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --
↪>
   <module name="scrubber" type="native">
    <input_file_name>scrubber.input</input_file_name>
    <input_file_path>/home/dealmeida/mac-dealmeida/gentoo-home/work/codes/
↪reprocessing/cortix-dev/input/</input_file_path>
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Ports have four types: 1) use, 2) provide, 3) input 4) output -->
    <!-- The *input*, and *output* types allow for module self-connection; see␣
↪network section  -->
    <port type="use"     mode="file.xml" multiplicity="1">inflow-gas</port>
    <port type="provide" mode="file.xml" multiplicity="1">off-gas</port>
   </module>

   <!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --
↪>
   <module name="filter" type="native">
    <input_file_name>filter.input</input_file_name>
    <input_file_path>/home/dealmeida/mac-dealmeida/gentoo-home/work/codes/
↪reprocessing/cortix-dev/input/</input_file_path>
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Ports have four types: 1) use, 2) provide, 3) input 4) output -->
    <!-- The *input*, and *output* types allow for module self-connection; see␣
↪network section  -->
    <port type="use"     mode="file.xml" multiplicity="1">inflow-gas</port>
    <port type="provide" mode="file.xml" multiplicity="1">off-gas</port>
   </module>

   <!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --
↪>
   <module name="offgas" type="native">
    <input_file_name>offgas.input</input_file_name>
    <input_file_path>/home/dealmeida/mac-dealmeida/gentoo-home/work/codes/
↪reprocessing/cortix-dev/input/</input_file_path>
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Ports have four types: 1) use, 2) provide, 3) input 4) output -->
```

(continues on next page)

(continued from previous page)

```
    <!-- The *input*, and *output* types allow for module self-connection; see␣
↪network section  -->
    <port type="use"     mode="file.xml" multiplicity="1">inflow-gas</port>
    <port type="provide" mode="file.xml" multiplicity="1">off-gas</port>
   </module>

   <!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --
↪>
   <module name="dissolverFpot" type="native">
    <library name='srnl-lib'>
     <parent_dir>
      /home/dealmeida/mac-dealmeida/gentoo-home/work/codes/reprocessing
     </parent_dir>
    </library>
    <input_file_name>dissolverFpot.input</input_file_name>
    <input_file_path>/home/dealmeida/mac-dealmeida/gentoo-home/work/codes/
↪reprocessing/cortix-dev/input/</input_file_path>
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Ports have four types: 1) use, 2) provide, 3) input 4) output -->
    <!-- The *input*, and *output* types allow for module self-connection; see␣
↪network section  -->
    <port type="input"   mode="file.xml"  multiplicity="1">dissolver-solo-input</port>
    <port type="use"     mode="file.xml"  multiplicity="1">solids-input</port>
    <port type="use"     mode="file.xml"  multiplicity="1">go-signal</port>
    <port type="use"     mode="file.xml"  multiplicity="1">fuel-bucket</port>
    <port type="use"     mode="file.xml"  multiplicity="1">condensate</port>
    <port type="provide" mode="file.xml"  multiplicity="1">status-signal</port>
    <port type="provide" mode="file.xml"  multiplicity="1">state</port>
    <port type="provide" mode="file.shlv" multiplicity="1">product</port>
    <port type="provide" mode="file.xml"  multiplicity="1">product-plot</port>
    <port type="provide" mode="file.shlv" multiplicity="1">solution</port>
    <port type="provide" mode="file.xml"  multiplicity="1">solution-plot</port>
    <port type="provide" mode="file.shlv" multiplicity="1">vapor</port>
    <port type="provide" mode="file.dat"  multiplicity="1">vapor-table</port>
    <port type="output"  mode="file.any"  multiplicity="1">persistent-output</port>
   </module>

   <!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --
↪>
   <module name="plume" type="wrapped">
    <library name='srnl-lib'>
     <parent_dir>
      /home/dealmeida/mac-dealmeida/gentoo-home/work/codes/reprocessing
     </parent_dir>
    </library>
    <input_file_name>plume.input</input_file_name>
    <input_file_path>/home/dealmeida/mac-dealmeida/gentoo-home/work/codes/
↪reprocessing/cortix-dev/input/</input_file_path>
    <executable_name>pfpl</executable_name>
    <executable_path>/home/dealmeida/mac-dealmeida/gentoo-home/work/codes/
↪reprocessing/facility_pfpl/</executable_path>
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
```

(continues on next page)

(continued from previous page)

```
    </logger>
    <!-- Ports have four types: 1) use, 2) provide, 3) input 4) output -->
    <!-- The *input*, and *output* types allow for module self-connection; see␣
↪network section  -->
    <port type="input"   mode="file.xml" multiplicity="1">plume-solo-input</port>
    <port type="use"     mode="file.xml" multiplicity="1">offgas-input</port>
    <port type="use"     mode="file.xml" multiplicity="1">offgas</port>
    <port type="provide" mode="file.xml" multiplicity="1">puff</port>
   </module>

   <!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --
↪>
   <module name="cooltower" type="wrapped">
    <library name='srnl-lib'>
     <parent_dir>
      /home/dealmeida/mac-dealmeida/gentoo-home/work/codes/reprocessing
     </parent_dir>
    </library>
    <input_file_name>cooltower.input</input_file_name>
    <input_file_path>/home/dealmeida/mac-dealmeida/gentoo-home/work/codes/
↪reprocessing/cortix-dev/input/</input_file_path>
    <executable_name>cttool.x</executable_name>
    <executable_path>/home/dealmeida/mac-dealmeida/gentoo-home/work/codes/
↪reprocessing/srnl-lib/cooltower/bin/</executable_path>
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Ports have four types: 1) use, 2) provide, 3) input 4) output -->
    <!-- The *input*, and *output* types allow for module self-connection; see␣
↪network section  -->
    <port type="input"   mode="file.xml" multiplicity="1">cooltower-solo-input</port>
    <port type="use"     mode="file.xml" multiplicity="1">water-input</port>
    <port type="use"     mode="file.xml" multiplicity="1">hot-water</port>
    <port type="provide" mode="file.xml" multiplicity="1">cold-water</port>
   </module>

   <!-- ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --
↪>
   <module name="modulib.pyplot" type="native">
    <library name='cortix'>
     <parent_dir>
      /home/dealmeida/mac-dealmeida/gentoo-home/work/codes/reprocessing/cortix-dev
     </parent_dir>
    </library>
    <input_file_name>pyplot.input</input_file_name>
    <input_file_path>/home/dealmeida/mac-dealmeida/gentoo-home/work/codes/
↪reprocessing/cortix-dev/input/</input_file_path>
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Ports have four types: 1) use, 2) provide, 3) input 4) output -->
    <!-- The *input*, and *output* types allow for module self-connection; see␣
↪network section  -->
    <port type="input" mode="file.xml" multiplicity="1">pyplot-solo-input</port>
    <port type="use"   mode="file.xml" multiplicity="1">time-sequence-input</port>
```

(continues on next page)

(continued from previous page)

```
    <port type="use"   mode="file.xml" multiplicity="1">time-sequence</port>
    <port type="use"   mode="file.xml" multiplicity="1">time-tables</port>
   </module>


   <!--                  Modules objects end here                              --
↪>
   <!-- :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: --
↪>

   <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
↪>
   <!--                  Network objects start here                            --
↪>

   <!-- Application: networks -->
   <!-- NB: each network has its own task definition above; with the same name -->
   <!-- NB: the connect tag is ordered: *from* is the receiver; *to* is the provider--
↪>
   <!-- NB: port labels are the "names" of the ports; "not" a file name necessarily-->
   <!-- NB: module labels must be composed with a "slot" number, say "name:x"-->

   <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
↪>
   <network name="solo-pyplot">
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Modules need to be given a slot number; use a colon after the name followed␣
↪by an integer-->
    <!-- For self-connection of a module: fromPort is use type; toPort is input type--
↪>
    <!-- For self-connection of a module: fromPort is output type; toPort is provide␣
↪type-->
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence-input"␣
↪toModuleSlot="modulib.pyplot:0" toPort="pyplot-solo-input"/>
   </network>


   <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
↪>
   <network name="solo-fueldepot">
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Modules need to be given a slot number; use a colon after the name followed␣
↪by an integer-->
    <!-- For self-connection of a module: fromPort is use type; toPort is input type--
↪>
    <!-- For self-connection of a module: fromPort is output type; toPort is provide␣
↪type-->
<!--
    <connect fromModuleSlot="fueldepot:0" fromPort="fuel-input" toModuleSlot=
↪"fueldepot:0" toPort="westinghouse-14x14"/>
    <connect fromModuleSlot="fueldepot:0" fromPort="persistent-output" toModuleSlot=
↪"fueldepot:0" toPort="fuel-bundle"/>
```

(continues on next page)

(continued from previous page)

```
-->
    <connect fromModuleSlot="fueldepot:0" fromPort="fuel-input" toModuleSlot=
↪"fueldepot:0" toPort="mark-31a"/>
    <connect fromModuleSlot="fueldepot:0" fromPort="persistent-output" toModuleSlot=
↪"fueldepot:0" toPort="fuel-bucket"/>
   </network>

   <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
↪>
   <network name="solo-shear">
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Modules need to be given a slot number; use a colon after the name followed␣
↪by an integer-->
    <!-- For self-connection of a module: fromPort is use type; toPort is input type--
↪>
    <!-- For self-connection of a module: fromPort is output type; toPort is provide␣
↪type-->
    <connect fromModuleSlot="oldchopper:0" fromPort="solids-input" toModuleSlot=
↪"oldchopper:0" toPort="Fuel_Solid"/>
    <connect fromModuleSlot="oldchopper:0" fromPort="gas-input" toModuleSlot=
↪"oldchopper:0" toPort="Gas_Release"/>
    <connect fromModuleSlot="oldchopper:0" fromPort="fines-input" toModuleSlot=
↪"oldchopper:0" toPort="Particulate"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"oldchopper:0" toPort="off-gas"/>
   </network>

   <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
↪>
   <network name="solo-fuel-accum">
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Modules need to be given a slot number; use a colon after the name followed␣
↪by an integer-->
    <!-- For self-connection of a module: fromPort is use type; toPort is input type--
↪>
    <!-- For self-connection of a module: fromPort is output type; toPort is provide␣
↪type-->
    <connect fromModuleSlot="storage:0" fromPort="solids" toModuleSlot="storage:0"␣
↪toPort="chopper-data-test"/>
    <connect fromModuleSlot="storage:0" fromPort="withdrawal-request" toModuleSlot=
↪"storage:0" toPort="dissolver-data-test"/>
   </network>

   <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
↪>
   <network name="solo-dissolverA">
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Modules need to be given a slot number; use a colon after the name followed␣
↪by an integer-->
```

(continues on next page)

(continued from previous page)

```
    <!-- For self-connection of a module: fromPort is use type; toPort is input type--
↪>
    <!-- For self-connection of a module: fromPort is output type; toPort is provide␣
↪type-->
    <connect fromModuleSlot="dissolverA:0" fromPort="solids-input" toModuleSlot=
↪"dissolverA:0" toPort="dissolver-solo-input"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"dissolverA:0" toPort="state"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"dissolverA:0" toPort="vapor"/>
   </network>

   <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
↪>
   <network name="solo-condenser">
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Modules need to be given a slot number; use a colon after the name followed␣
↪by an integer-->
    <!-- For self-connection of a module: fromPort is use type; toPort is input type--
↪>
    <!-- For self-connection of a module: fromPort is output type; toPort is provide␣
↪type-->
    <connect fromModuleSlot="condenser:0" fromPort="vapor" toModuleSlot="condenser:0"␣
↪toPort="condenser-solo-input"/>
    <connect fromModuleSlot="condenser:0" fromPort="persistent-output" toModuleSlot=
↪"condenser:0" toPort="condensate"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"condenser:0" toPort="state"/>
   </network>

   <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
↪>
   <network name="solo-tank">
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Modules need to be given a slot number; use a colon after the name followed␣
↪by an integer-->
    <!-- For self-connection of a module: fromPort is use type; toPort is input type--
↪>
    <!-- For self-connection of a module: fromPort is output type; toPort is provide␣
↪type-->
    <connect fromModuleSlot="tank:0" fromPort="liquid" toModuleSlot="tank:0" toPort=
↪"tank-solo-input"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"tank:0" toPort="state"/>
   </network>

   <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
↪>
   <network name="solo-cooltower">
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
```

(continues on next page)

(continued from previous page)

```
    <console_handler level="INFO"> </console_handler>
   </logger>
   <!-- Modules need to be given a slot number; use a colon after the name followed␣
↪by an integer-->
   <!-- For self-connection of a module: fromPort is use type; toPort is input type--
↪>
   <!-- For self-connection of a module: fromPort is output type; toPort is provide␣
↪type-->
   <connect fromModuleSlot="cooltower:0" fromPort="water-input" toModuleSlot=
↪"cooltower:0" toPort="cooltower-solo-input"/>
   <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"cooltower:0" toPort="cold-water"/>
  </network>

  <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
↪>
  <network name="solo-plume">
   <logger level="DEBUG">
    <file_handler level="DEBUG"> </file_handler>
    <console_handler level="INFO"> </console_handler>
   </logger>
   <!-- Modules need to be given a slot number; use a colon after the name followed␣
↪by an integer-->
   <!-- For self-connection of a module: fromPort is use type; toPort is input type--
↪>
   <!-- For self-connection of a module: fromPort is output type; toPort is provide␣
↪type-->
   <connect fromModuleSlot="plume:0" fromPort="offgas-input" toModuleSlot="plume:0"␣
↪toPort="plume-solo-input"/>
   <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-tables" toModuleSlot=
↪"plume:0" toPort="puff"/>
  </network>

  <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
↪>
  <network name="solo-solventxtract">
   <logger level="DEBUG">
    <file_handler level="DEBUG"> </file_handler>
    <console_handler level="INFO"> </console_handler>
   </logger>
   <!-- Modules need to be given a slot number; use a colon after the name followed␣
↪by an integer-->
   <!-- For self-connection of a module: fromPort is use type; toPort is input type--
↪>
   <!-- For self-connection of a module: fromPort is output type; toPort is provide␣
↪type-->
   <connect fromModuleSlot="solventxtract:0" fromPort="liquid-input" toModuleSlot=
↪"solventxtract:0" toPort="solventxtract-solo-input"/>
   <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"solventxtract:0" toPort="status-signal"/>
   <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"solventxtract:0" toPort="state"/>
<!--    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence"␣
↪toModuleSlot="solventxtract:0" toPort="product-plot"/> -->
  </network>

  <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
↪>
```

(continues on next page)

(continued from previous page)

```
    <network name="fueldepot-chopper">
     <logger level="DEBUG">
      <file_handler level="DEBUG"> </file_handler>
      <console_handler level="INFO"> </console_handler>
     </logger>
     <!-- Modules need to be given a slot number; use a colon after the name followed␣
↪by an integer-->
     <!-- For self-connection of a module: fromPort is use type; toPort is input type--
↪>
     <!-- For self-connection of a module: fromPort is output type; toPort is provide␣
↪type-->
     <connect fromModuleSlot="fueldepot:0" fromPort="fuel-input" toModuleSlot=
↪"fueldepot:0" toPort="westinghouse-14x14"/>
     <connect fromModuleSlot="fueldepot:0" fromPort="go-signal" toModuleSlot="chopper:0
↪" toPort="status-signal"/>
     <connect fromModuleSlot="chopper:0" fromPort="fuel-bundle" toModuleSlot=
↪"fueldepot:0" toPort="fuel-bundle"/>
<!--
     <connect fromModuleSlot="chopper:0" fromPort="persistent-output" toModuleSlot=
↪"chopper:0" toPort="fuel-segments"/>
-->
     <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"chopper:0" toPort="state"/>
     <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"chopper:0" toPort="status-signal"/>
    </network>

    <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
↪>
    <network name="fueldepot-chopper-storage">
     <logger level="DEBUG">
      <file_handler level="DEBUG"> </file_handler>
      <console_handler level="INFO"> </console_handler>
     </logger>
     <!-- Modules need to be given a slot number; use a colon after the name followed␣
↪by an integer-->
     <!-- For self-connection of a module: fromPort is use type; toPort is input type--
↪>
     <!-- For self-connection of a module: fromPort is output type; toPort is provide␣
↪type-->
     <connect fromModuleSlot="fueldepot:0" fromPort="fuel-input" toModuleSlot=
↪"fueldepot:0" toPort="westinghouse-14x14"/>
     <connect fromModuleSlot="fueldepot:0" fromPort="go-signal" toModuleSlot="chopper:0
↪" toPort="status-signal"/>
     <connect fromModuleSlot="chopper:0" fromPort="fuel-bundle" toModuleSlot=
↪"fueldepot:0" toPort="fuel-bundle"/>
     <connect fromModuleSlot="chopper:0" fromPort="go-signal" toModuleSlot="storage:0"␣
↪toPort="status-signal"/>
     <connect fromModuleSlot="storage:0" fromPort="fuel-segments" toModuleSlot=
↪"chopper:0" toPort="fuel-segments"/>
     <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"chopper:0" toPort="state"/>
     <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"chopper:0" toPort="status-signal"/>
     <connect fromModuleSlot="modulib.pyplot:1" fromPort="time-sequence" toModuleSlot=
↪"storage:0" toPort="state"/>
     <connect fromModuleSlot="modulib.pyplot:1" fromPort="time-sequence" toModuleSlot=
↪"storage:0" toPort="off-gas"/>
```

(continues on next page)

(continued from previous page)

```
    <connect fromModuleSlot="modulib.pyplot:1" fromPort="time-sequence" toModuleSlot=
→"storage:0" toPort="status-signal"/>
   </network>

   <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
→>
   <network name="fueldepot-chopper-dissolver">
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Modules need to be given a slot number; use a colon after the name followed␣
→by an integer-->
    <!-- For self-connection of a module: fromPort is use type; toPort is input type--
→>
    <!-- For self-connection of a module: fromPort is output type; toPort is provide␣
→type-->
    <connect fromModuleSlot="fueldepot:0" fromPort="fuel-input" toModuleSlot=
→"fueldepot:0" toPort="westinghouse-14x14"/>
    <connect fromModuleSlot="fueldepot:0" fromPort="go-signal" toModuleSlot="chopper:0
→" toPort="status-signal"/>
    <connect fromModuleSlot="chopper:0" fromPort="fuel-bundle" toModuleSlot=
→"fueldepot:0" toPort="fuel-bundle"/>
    <connect fromModuleSlot="chopper:0" fromPort="go-signal" toModuleSlot="dissolver:0
→" toPort="status-signal"/>
    <connect fromModuleSlot="dissolver:0" fromPort="fuel-segments" toModuleSlot=
→"chopper:0" toPort="fuel-segments"/>
    <connect fromModuleSlot="dissolver:0" fromPort="persistent-output" toModuleSlot=
→"dissolver:0" toPort="vapor-table"/>
    <connect fromModuleSlot="dissolver:0" fromPort="persistent-output" toModuleSlot=
→"dissolver:0" toPort="vapor"/>
    <connect fromModuleSlot="dissolver:0" fromPort="persistent-output" toModuleSlot=
→"dissolver:0" toPort="product"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
→"chopper:0" toPort="state"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
→"chopper:0" toPort="status-signal"/>
    <connect fromModuleSlot="modulib.pyplot:1" fromPort="time-sequence" toModuleSlot=
→"dissolver:0" toPort="status-signal"/>
    <connect fromModuleSlot="modulib.pyplot:1" fromPort="time-sequence" toModuleSlot=
→"dissolver:0" toPort="state"/>
<!--    <connect fromModuleSlot="modulib.pyplot:1" fromPort="time-sequence"␣
→toModuleSlot="dissolver:0" toPort="product-plot"/>  -->
   </network>

   <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
→>
   <network name="fueldepot-dissolverFpot">
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Modules need to be given a slot number; use a colon after the name followed␣
→by an integer-->
    <!-- For self-connection of a module: fromPort is use type; toPort is input type--
→>
    <!-- For self-connection of a module: fromPort is output type; toPort is provide␣
→type-->
```

(continues on next page)

(continued from previous page)

```
    <connect fromModuleSlot="fueldepot:0" fromPort="fuel-input" toModuleSlot=
↪"fueldepot:0" toPort="mark-31a"/>
    <connect fromModuleSlot="fueldepot:0" fromPort="go-signal" toModuleSlot=
↪"dissolverFpot:0" toPort="status-signal"/>
    <connect fromModuleSlot="dissolverFpot:0" fromPort="fuel-bucket" toModuleSlot=
↪"fueldepot:0" toPort="fuel-bucket"/>
    <connect fromModuleSlot="dissolverFpot:0" fromPort="persistent-output"␣
↪toModuleSlot="dissolverFpot:0" toPort="vapor-table"/>
    <connect fromModuleSlot="dissolverFpot:0" fromPort="persistent-output"␣
↪toModuleSlot="dissolverFpot:0" toPort="vapor"/>
<!--
    <connect fromModuleSlot="dissolverFpot:0" fromPort="persistent-output"␣
↪toModuleSlot="dissolverFpot:0" toPort="solution"/>
-->
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"dissolverFpot:0" toPort="status-signal"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"dissolverFpot:0" toPort="state"/>
<!--
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"dissolverFpot:0" toPort="solution-plot"/>
-->
   </network>

   <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
↪>
   <network name="tank-feedprep">
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Modules need to be given a slot number; use a colon after the name followed␣
↪by an integer-->
    <!-- For self-connection of a module: fromPort is use type; toPort is input type--
↪>
    <!-- For self-connection of a module: fromPort is output type; toPort is t type-->
    <connect fromModuleSlot="tank:0" fromPort="liquid" toModuleSlot="tank:0" toPort=
↪"tank-solo-input"/>
    <connect fromModuleSlot="tank:0" fromPort="go-signal" toModuleSlot="feedprep:0"␣
↪toPort="status-signal"/>
    <connect fromModuleSlot="feedprep:0" fromPort="liquid" toModuleSlot="tank:0"␣
↪toPort="product"/>
    <connect fromModuleSlot="feedprep:0" fromPort="persistent-output" toModuleSlot=
↪"feedprep:0" toPort="product"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"tank:0" toPort="state"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"tank:0" toPort="status-signal"/>
    <connect fromModuleSlot="modulib.pyplot:1" fromPort="time-sequence" toModuleSlot=
↪"feedprep:0" toPort="state"/>
    <connect fromModuleSlot="modulib.pyplot:1" fromPort="time-sequence" toModuleSlot=
↪"feedprep:0" toPort="status-signal"/>
   </network>

   <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
↪>
   <network name="fueldepot-chopper-dissolver-tank">
```

(continues on next page)

(continued from previous page)

```
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Modules need to be given a slot number; use a colon after the name followed␣
↪by an integer-->
    <!-- For self-connection of a module: fromPort is use type; toPort is input type--
↪>
    <!-- For self-connection of a module: fromPort is output type; toPort is provide␣
↪type-->
    <connect fromModuleSlot="fueldepot:0" fromPort="fuel-input" toModuleSlot=
↪"fueldepot:0" toPort="westinghouse-14x14"/>
    <connect fromModuleSlot="fueldepot:0" fromPort="go-signal" toModuleSlot="chopper:0
↪" toPort="status-signal"/>
    <connect fromModuleSlot="chopper:0" fromPort="fuel-bundle" toModuleSlot=
↪"fueldepot:0" toPort="fuel-bundle"/>
    <connect fromModuleSlot="chopper:0" fromPort="go-signal" toModuleSlot="dissolver:0
↪" toPort="status-signal"/>
    <connect fromModuleSlot="dissolver:0" fromPort="fuel-segments" toModuleSlot=
↪"chopper:0" toPort="fuel-segments"/>
    <connect fromModuleSlot="dissolver:0" fromPort="persistent-output" toModuleSlot=
↪"dissolver:0" toPort="vapor-table"/>
    <connect fromModuleSlot="dissolver:0" fromPort="persistent-output" toModuleSlot=
↪"dissolver:0" toPort="vapor"/>
    <connect fromModuleSlot="dissolver:0" fromPort="go-signal" toModuleSlot="tank:0"␣
↪toPort="status-signal"/>
    <connect fromModuleSlot="tank:0" fromPort="liquid" toModuleSlot="dissolver:0"␣
↪toPort="product"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"chopper:0" toPort="state"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"chopper:0" toPort="status-signal"/>
    <connect fromModuleSlot="modulib.pyplot:1" fromPort="time-sequence" toModuleSlot=
↪"dissolver:0" toPort="status-signal"/>
    <connect fromModuleSlot="modulib.pyplot:1" fromPort="time-sequence" toModuleSlot=
↪"dissolver:0" toPort="state"/>
<!--    <connect fromModuleSlot="modulib.pyplot:1" fromPort="time-sequence"␣
↪toModuleSlot="dissolver:0" toPort="product-plot"/>  -->
    <connect fromModuleSlot="modulib.pyplot:2" fromPort="time-sequence" toModuleSlot=
↪"tank:0" toPort="state"/>
    <connect fromModuleSlot="modulib.pyplot:2" fromPort="time-sequence" toModuleSlot=
↪"tank:0" toPort="status-signal"/>
   </network>

   <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
↪>
   <network name="fueldepot-chopper-dissolver-tank-feedprep">
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Modules need to be given a slot number; use a colon after the name followed␣
↪by an integer-->
    <!-- For self-connection of a module: fromPort is use type; toPort is input type--
↪>
    <!-- For self-connection of a module: fromPort is output type; toPort is t type-->
    <connect fromModuleSlot="fueldepot:0" fromPort="fuel-input" toModuleSlot=
↪"fueldepot:0" toPort="westinghouse-14x14"/>
```

(continues on next page)

(continued from previous page)

```
    <connect fromModuleSlot="fueldepot:0" fromPort="go-signal" toModuleSlot="chopper:0
↪" toPort="status-signal"/>
    <connect fromModuleSlot="chopper:0" fromPort="fuel-bundle" toModuleSlot=
↪"fueldepot:0" toPort="fuel-bundle"/>
    <connect fromModuleSlot="chopper:0" fromPort="go-signal" toModuleSlot="dissolver:0
↪" toPort="status-signal"/>
    <connect fromModuleSlot="dissolver:0" fromPort="fuel-segments" toModuleSlot=
↪"chopper:0" toPort="fuel-segments"/>
    <connect fromModuleSlot="dissolver:0" fromPort="persistent-output" toModuleSlot=
↪"dissolver:0" toPort="vapor-table"/>
    <connect fromModuleSlot="dissolver:0" fromPort="persistent-output" toModuleSlot=
↪"dissolver:0" toPort="vapor"/>
    <connect fromModuleSlot="dissolver:0" fromPort="go-signal" toModuleSlot="tank:0"␣
↪toPort="status-signal"/>
    <connect fromModuleSlot="tank:0" fromPort="liquid" toModuleSlot="dissolver:0"␣
↪toPort="product"/>
    <connect fromModuleSlot="tank:0" fromPort="go-signal" toModuleSlot="feedprep:0"␣
↪toPort="status-signal"/>
    <connect fromModuleSlot="feedprep:0" fromPort="liquid" toModuleSlot="tank:0"␣
↪toPort="product"/>
    <connect fromModuleSlot="feedprep:0" fromPort="persistent-output" toModuleSlot=
↪"feedprep:0" toPort="product"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"chopper:0" toPort="state"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"chopper:0" toPort="status-signal"/>
    <connect fromModuleSlot="modulib.pyplot:1" fromPort="time-sequence" toModuleSlot=
↪"dissolver:0" toPort="status-signal"/>
    <connect fromModuleSlot="modulib.pyplot:1" fromPort="time-sequence" toModuleSlot=
↪"dissolver:0" toPort="state"/>
    <connect fromModuleSlot="modulib.pyplot:2" fromPort="time-sequence" toModuleSlot=
↪"tank:0" toPort="state"/>
    <connect fromModuleSlot="modulib.pyplot:2" fromPort="time-sequence" toModuleSlot=
↪"tank:0" toPort="status-signal"/>
    <connect fromModuleSlot="modulib.pyplot:3" fromPort="time-sequence" toModuleSlot=
↪"feedprep:0" toPort="state"/>
<!--    <connect fromModuleSlot="modulib.pyplot:3" fromPort="time-sequence"␣
↪toModuleSlot="feedprep:0" toPort="product-plot"/> -->
    <connect fromModuleSlot="modulib.pyplot:3" fromPort="time-sequence" toModuleSlot=
↪"feedprep:0" toPort="status-signal"/>
   </network>

   <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
↪>
   <network name="fueldepot-chopper-dissolver-tank-feedprep-solventxtract">
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Modules need to be given a slot number; use a colon after the name followed␣
↪by an integer-->
    <!-- For self-connection of a module: fromPort is use type; toPort is input type--
↪>
    <!-- For self-connection of a module: fromPort is output type; toPort is provide␣
↪type-->
    <connect fromModuleSlot="fueldepot:0" fromPort="fuel-input" toModuleSlot=
↪"fueldepot:0" toPort="westinghouse-14x14"/>
```

(continues on next page)

(continued from previous page)

```
    <connect fromModuleSlot="fueldepot:0" fromPort="go-signal" toModuleSlot="chopper:0
↪" toPort="status-signal"/>
    <connect fromModuleSlot="chopper:0" fromPort="fuel-bundle" toModuleSlot=
↪"fueldepot:0" toPort="fuel-bundle"/>
    <connect fromModuleSlot="chopper:0" fromPort="go-signal" toModuleSlot="dissolver:0
↪" toPort="status-signal"/>
    <connect fromModuleSlot="dissolver:0" fromPort="fuel-segments" toModuleSlot=
↪"chopper:0" toPort="fuel-segments"/>
    <connect fromModuleSlot="dissolver:0" fromPort="go-signal" toModuleSlot="tank:0"␣
↪toPort="status-signal"/>
    <connect fromModuleSlot="tank:0" fromPort="liquid" toModuleSlot="dissolver:0"␣
↪toPort="product"/>
    <connect fromModuleSlot="tank:0" fromPort="go-signal" toModuleSlot="feedprep:0"␣
↪toPort="status-signal"/>
    <connect fromModuleSlot="feedprep:0" fromPort="liquid" toModuleSlot="tank:0"␣
↪toPort="product"/>
    <connect fromModuleSlot="feedprep:0" fromPort="go-signal" toModuleSlot=
↪"solventxtract:0" toPort="status-signal"/>
    <connect fromModuleSlot="solventxtract:0" fromPort="liquid" toModuleSlot=
↪"feedprep:0" toPort="product"/>
<!--    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence"␣
↪toModuleSlot="chopper:0" toPort="state"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"chopper:0" toPort="status-signal"/> -->
    <connect fromModuleSlot="modulib.pyplot:1" fromPort="time-sequence" toModuleSlot=
↪"dissolver:0" toPort="status-signal"/>
    <connect fromModuleSlot="modulib.pyplot:1" fromPort="time-sequence" toModuleSlot=
↪"dissolver:0" toPort="state"/>
<!--    <connect fromModuleSlot="modulib.pyplot:1" fromPort="time-sequence"␣
↪toModuleSlot="dissolver:0" toPort="product-plot"/>  -->
<!--    <connect fromModuleSlot="modulib.pyplot:2" fromPort="time-sequence"␣
↪toModuleSlot="tank:0" toPort="state"/>
    <connect fromModuleSlot="modulib.pyplot:2" fromPort="time-sequence" toModuleSlot=
↪"tank:0" toPort="status-signal"/>  -->
    <connect fromModuleSlot="modulib.pyplot:2" fromPort="time-sequence" toModuleSlot=
↪"feedprep:0" toPort="state"/>
    <connect fromModuleSlot="modulib.pyplot:2" fromPort="time-sequence" toModuleSlot=
↪"feedprep:0" toPort="status-signal"/>
    <connect fromModuleSlot="modulib.pyplot:3" fromPort="time-sequence" toModuleSlot=
↪"solventxtract:0" toPort="status-signal"/>
   </network>

   <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
↪>
   <network name="shear-dissolve-offgas">
   <!-- Network name must match a task name-->
    <logger level="DEBUG">
     <file_handler level="DEBUG"> </file_handler>
     <console_handler level="INFO"> </console_handler>
    </logger>
    <!-- Modules need to be given a slot number; use a colon after the name followed␣
↪by an integer-->
    <!-- For self-connection of a module: fromPort is use type; toPort is input type--
↪>
    <!-- For self-connection of a module: fromPort is output type; toPort is provide␣
↪type-->
    <connect fromModuleSlot="oldchopper:0" fromPort="solids-input" toModuleSlot=
↪"oldchopper:0" toPort="Fuel_Solid"/>
```

(continues on next page)

(continued from previous page)

```
    <connect fromModuleSlot="oldchopper:0" fromPort="gas-input" toModuleSlot=
↪"oldchopper:0" toPort="Gas_Release"/>
    <connect fromModuleSlot="oldchopper:0" fromPort="fines-input" toModuleSlot=
↪"oldchopper:0" toPort="Particulate"/>
    <connect fromModuleSlot="storage:0" fromPort="solids" toModuleSlot="oldchopper:0"␣
↪toPort="solids"/>
    <connect fromModuleSlot="storage:0" fromPort="withdrawal-request" toModuleSlot=
↪"dissolverA:0" toPort="signal"/>
    <connect fromModuleSlot="dissolverA:0" fromPort="solids" toModuleSlot="storage:0"␣
↪toPort="fuel-segments"/>
    <connect fromModuleSlot="condenser:0" fromPort="vapor" toModuleSlot="dissolverA:0
↪" toPort="vapor"/>
    <connect fromModuleSlot="dissolverA:0" fromPort="condensate" toModuleSlot=
↪"condenser:0" toPort="condensate"/>
    <connect fromModuleSlot="scrubber:0" fromPort="inflow-gas" toModuleSlot=
↪"oldchopper:0" toPort="off-gas"/>
    <connect fromModuleSlot="scrubber:0" fromPort="inflow-gas" toModuleSlot="storage:0
↪" toPort="off-gas"/>
    <connect fromModuleSlot="scrubber:0" fromPort="inflow-gas" toModuleSlot=
↪"condenser:0" toPort="off-gas"/>
    <connect fromModuleSlot="filter:0" fromPort="inflow-gas" toModuleSlot="scrubber:0
↪" toPort="off-gas"/>
    <connect fromModuleSlot="offgas:0" fromPort="inflow-gas" toModuleSlot="filter:0"␣
↪toPort="off-gas"/>
<!--
    <connect fromModuleSlot="plume" fromPort="off-gas" toModuleSlot="offgas" toPort=
↪"off-gas"/>
-->
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"oldchopper:0" toPort="off-gas"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"storage:0" toPort="off-gas"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"storage:0" toPort="state"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"dissolverA:0" toPort="signal"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"dissolverA:0" toPort="vapor"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"dissolverA:0" toPort="state"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"condenser:0" toPort="off-gas"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"condenser:0" toPort="condensate"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"scrubber:0" toPort="off-gas"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"filter:0" toPort="off-gas"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"offgas:0" toPort="off-gas"/>
<!--
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-tables" toModuleSlot=
↪"plume:0" toPort="time-puff"/>
-->
   </network>

   <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
↪>
```

(continues on next page)

(continued from previous page)

```
<network name="shear-dissolve">
<!-- Network name must match a task name-->
 <logger level="DEBUG">
  <file_handler level="DEBUG"> </file_handler>
  <console_handler level="INFO"> </console_handler>
 </logger>
 <!-- Modules need to be given a slot number; use a colon after the name followed␣
↪by an integer-->
 <!-- For self-connection of a module: fromPort is use type; toPort is input type--
↪>
 <!-- For self-connection of a module: fromPort is output type; toPort is provide␣
↪type-->
 <connect fromModuleSlot="chopper:0" fromPort="solids-input" toModuleSlot=
↪"chopper:0" toPort="Fuel_Solid"/>
 <connect fromModuleSlot="chopper:0" fromPort="gas-input" toModuleSlot="chopper:0"␣
↪toPort="Gas_Release"/>
 <connect fromModuleSlot="chopper:0" fromPort="fines-input" toModuleSlot="chopper:0
↪" toPort="Particulate"/>
 <connect fromModuleSlot="storage:0" fromPort="solids" toModuleSlot="chopper:0"␣
↪toPort="solids"/>
 <connect fromModuleSlot="storage:0" fromPort="withdrawal-request" toModuleSlot=
↪"dissolverA:0" toPort="signal"/>
 <connect fromModuleSlot="dissolverA:0" fromPort="solids" toModuleSlot="storage:0"␣
↪toPort="fuel-segments"/>
 <connect fromModuleSlot="condenser:0" fromPort="vapor" toModuleSlot="dissolverA:0
↪" toPort="vapor"/>
 <connect fromModuleSlot="dissolverA:0" fromPort="condensate" toModuleSlot=
↪"condenser:0" toPort="condensate"/>
 <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"chopper:0" toPort="off-gas"/>
 <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"storage:0" toPort="off-gas"/>
 <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"storage:0" toPort="state"/>
 <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"dissolverA:0" toPort="signal"/>
 <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"dissolverA:0" toPort="vapor"/>
 <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"dissolverA:0" toPort="state"/>
 <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"condenser:0" toPort="off-gas"/>
 <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"condenser:0" toPort="condensate"/>
</network>

<!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
↪>
<network name="shear-double-dissolve-single-condense">
<!-- Network name must match a task name-->
 <logger level="DEBUG">
  <file_handler level="DEBUG"> </file_handler>
  <console_handler level="INFO"> </console_handler>
 </logger>
 <!-- Modules need to be given a slot number; use a colon after the name followed␣
↪by an integer-->
 <!-- For self-connection of a module: fromPort is use type; toPort is input type--
↪>
```

(continues on next page)

(continued from previous page)

```
    <!-- For self-connection of a module: fromPort is output type; toPort is provide␣
↪type-->
    <connect fromModuleSlot="chopper:0" fromPort="solids-input" toModuleSlot=
↪"chopper:0" toPort="Fuel_Solid"/>
    <connect fromModuleSlot="chopper:0" fromPort="gas-input" toModuleSlot="chopper:0"␣
↪toPort="Gas_Release"/>
    <connect fromModuleSlot="chopper:0" fromPort="fines-input" toModuleSlot="chopper:0
↪" toPort="Particulate"/>
    <connect fromModuleSlot="storage:0" fromPort="solids" toModuleSlot="chopper:0"␣
↪toPort="solids"/>
    <connect fromModuleSlot="storage:0" fromPort="withdrawal-request" toModuleSlot=
↪"dissolverA:0" toPort="signal"/>
    <connect fromModuleSlot="storage:0" fromPort="withdrawal-request" toModuleSlot=
↪"dissolverA:1" toPort="signal"/>
    <connect fromModuleSlot="dissolverA:0" fromPort="solids" toModuleSlot="storage:0"␣
↪toPort="fuel-segments"/>
    <connect fromModuleSlot="dissolverA:1" fromPort="solids" toModuleSlot="storage:0"␣
↪toPort="fuel-segments"/>
    <connect fromModuleSlot="condenser:0" fromPort="vapor" toModuleSlot="dissolverA:0
↪" toPort="vapor"/>
    <connect fromModuleSlot="dissolverA:0" fromPort="condensate" toModuleSlot=
↪"condenser:0" toPort="condensate"/>
    <connect fromModuleSlot="condenser:0" fromPort="vapor" toModuleSlot="dissolverA:1
↪" toPort="vapor"/>
    <connect fromModuleSlot="dissolverA:1" fromPort="condensate" toModuleSlot=
↪"condenser:0" toPort="condensate"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"chopper:0" toPort="off-gas"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"storage:0" toPort="off-gas"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"storage:0" toPort="state"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"dissolverA:0" toPort="signal"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"dissolverA:0" toPort="vapor"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"dissolverA:0" toPort="state"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"dissolverA:1" toPort="signal"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"dissolverA:1" toPort="vapor"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"dissolverA:1" toPort="state"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"condenser:0" toPort="off-gas"/>
    <connect fromModuleSlot="modulib.pyplot:0" fromPort="time-sequence" toModuleSlot=
↪"condenser:0" toPort="condensate"/>
   </network>

   <!--                  Network objects end here                                  --
↪>
   <!-- o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o --
↪>

  <!--                   Application object ends here                              --
↪>
```

(continues on next page)

(continued from previous page)

```
  <!-- ############################################################################ --
↪>
  </application>

 <!--                Simulation object ends here                                  --
↪>
 <!-- ============================================================================ --
↪>
 </simulation>

</cortix_config>
```

## 3.3 modulib

### 3.3.1 droplet

#### cortix_driver

Cortix driver for the PyPlot module.

**class** cortix_driver.**CortixDriver**(*slot_id*, *input_full_path_file_name*, *exec_full_path_file_name*, *work_dir*, *ports=[]*, *cortix_start_time=0.0*, *cortix_final_time=0.0*, *cortix_time_step=0.0*, *cortix_time_unit=None*)

Bases: `object`

**call_ports**(*cortix_time=0.0*)
Call all ports at cortix_time

**execute**(*cortix_time=0.0*, *time_step=0.0*)
Evolve system from cortix_time to cortix_time + time_step

#### droplet

Droplet module example in Cortix.

**class** droplet.**Droplet**(*slot_id*, *input_full_path_file_name*, *work_dir*, *ports=[]*, *cortix_start_time=0.0*, *cortix_final_time=0.0*, *cortix_time_step=0.0*, *cortix_time_unit=None*)

Bases: `object`

Droplet module used example in Cortix.

**_Droplet__evolve**(*cortix_time=0.0*, *cortix_time_step=0.0*)
ODE IVP problem: Given the initial data at $t = 0$, $u_1(0) = x_0$, $u_2(0) = v_0 = \dot{u}_1(0)$, solve $\frac{\mathrm{d}u}{\mathrm{d}t} = f(u)$ in the interval $0 \le t \le t_f$. When $u_1(t)$ is negative, bounce the droplet to a random height between 0 and $1.2\,x_0$ with no velocity, and continue the time integration until $t = t_f$.

**_Droplet__provide_output**(*port_file*, *at_time*)
Provide data that will remain in disk after the simulation ends.

**call_ports**(*cortix_time=0.0*)
Transfer data at cortix_time

**execute** (*cortix_time=0.0*, *cortix_time_step=0.0*)
Evolve system from cortix_time to cortix_time + cortix_time_step

## 3.3.2 wind

### cortix_driver

Cortix driver for the PyPlot module.

**class** cortix_driver.**CortixDriver** (*slot_id*, *input_full_path_file_name*, *exec_full_path_file_name*, *work_dir*, *ports=[]*, *cortix_start_time=0.0*, *cortix_final_time=0.0*, *cortix_time_step=0.0*, *cortix_time_unit=None*)
Bases: object

**call_ports** (*cortix_time=0.0*)
Call all ports at cortix_time

**execute** (*cortix_time=0.0*, *time_step=0.0*)
Evolve system from cortix_time to cortix_time + time_step

### wind

Wind module example in Cortix.

**class** wind.**Wind** (*slot_id*, *input_full_path_file_name*, *work_dir*, *ports=[]*, *cortix_start_time=0.0*, *cortix_final_time=0.0*, *cortix_time_step=0.0*, *cortix_time_unit=None*)
Bases: object

Wind module used example in Cortix.

**call_ports** (*cortix_time=0.0*)
Transfer data at cortix_time

**execute** (*cortix_time=0.0*, *cortix_time_step=0.0*)
Evolve system from cortix_time to cortix_time + cortix_time_step

CHAPTER

# FOUR

# SUPPORT

## 4.1 actor

This is a simple way to hide the name of species of interest in a simulation. The user would modify and copy this class into the Cortix module of interest and keep it private.

Author: Valmor de Almeida dealmeidav@ornl.gov; vfda Sat Aug 15 13:41:12 EDT 2015

**class** actor.**Actor**(*name*)

Bases: object

See atoms list in Specie.

**atoms**

**formula**

## 4.2 fuel_bucket

This FuelBucket class is a container for usage with other plant-level process modules. It is meant to represent a fuel bucket of a metal fuel reactor. ——- ATTENTION: ——- This container uses Phase() for phases (cladding and fuel). Therefore user is responsible to make the "history" of the phases consistent. See Phase() info.

Author: Valmor de Almeida dealmeidav@ornl.gov; vfda

**class** fuel_bucket.**FuelBucket**(*specs=Empty DataFrame Columns: [] Index: []*)

Bases: object

**cladding_end_thickness**

**cladding_mass**

**cladding_phase**

**cladding_volume**

**cladding_wall_thickness**

**fresh_u235_mass**

**fresh_u238_mass**

**fresh_u_mass**

**fuel_enrichment**

**fuel_mass**

**fuel_mass_unit**

**fuel_phase**

**fuel_radioactivity**

**fuel_volume**

**gamma_pwr**

**get_cladding_end_thickness**()

**get_cladding_mass**()

**get_cladding_phase**()

**get_cladding_volume**()

**get_cladding_wall_thickness**()

**get_fresh_u235_mass**()

**get_fresh_u238_mass**()

**get_fresh_u_mass**()

**get_fuel_enrichment**()

**get_fuel_mass**()

**get_fuel_mass_unit**()

**get_fuel_phase**()

**get_fuel_radioactivity**()

**get_fuel_volume**()

**get_gamma_pwr**()

**get_heat_pwr**()

**get_inner_slug_id**()

**get_inner_slug_od**()

**get_n_slugs**()

**get_name**()

**get_outer_slug_id**()

**get_outer_slug_od**()

**get_radioactivity**()

**get_slug_cladding_volume**()

**get_slug_fuel_volume**()

**get_slug_length**()

**get_slug_type**()

**heat_pwr**

**inner_slug_id**

**inner_slug_od**

**n_slugs**

**name**

**outer_slug_id**

**outer_slug_od**

**radioactivity**

**set_cladding_phase** (*phase*)

**set_fuel_phase** (*phase*)

**set_slug_length** (*x*)

**slug_cladding_volume**

**slug_fuel_volume**

**slug_length**

**slug_type**

## 4.3 fuel_bundle

This FuelBundle class is a container for usage with other plant-level process modules. It is meant to represent a fuel bundle of an oxide fuel LWR reactor. There are three main data structures:

1. fuel bundle specs
2. solid phase
3. gas phase

The container user will have to provide all the data and from then on, this class will help acess the data. The printing methods reveal the contained data.

Author: Valmor de Almeida dealmeidav@ornl.gov; vfda Sun Dec 27 15:06:55 EST 2015

**class** fuel_bundle.**FuelBundle** (*specs=Empty DataFrame Columns: [] Index: []*)
Bases: object

**fresh_u235_mass**

**fresh_u238_mass**

**fresh_u_mass**

**fuel_enrichment**

**fuel_mass**

**fuel_mass_unit**

**fuel_pin_length**

**fuel_pin_radius**

**fuel_pin_volume**

**fuel_radioactivity**

**fuel_rod_od**

**fuel_volume**

**gamma_pwr**

**gas_mass**

**gas_phase**

**gas_radioactivity**

**get_fresh_U235_mass**()

**get_fresh_u238_mass**()

**get_fresh_u_mass**()

**get_fuel_enrichment**()

**get_fuel_mass**()

**get_fuel_mass_unit**()

**get_fuel_pin_length**()

**get_fuel_pin_radius**()

**get_fuel_pin_volume**()

**get_fuel_radioactivity**()

**get_fuel_rod_od**()

**get_fuel_volume**()

**get_gamma_pwr**()

**get_gas_mass**()

**get_gas_phase**()

**get_gas_radioactivity**()

**get_heat_pwr**()

**get_n_fuel_rods**()

**get_name**()

**get_radioactivity**()

**get_solid_phase**()

**heat_pwr**

**n_fuel_rods**

**name**

**radioactivity**

**set_fuel_pin_length**(*x*)

**set_gas_phase**(*phase*)

**set_solid_phase**(*phase*)

**solid_phase**

## 4.4 fuel_segment

Fuel segment Author: Valmor de Almeida dealmeidav@ornl.gov; vfda Sat Jun 27 14:46:49 EDT 2015

**class** fuel_segment.**FuelSegment**(*geometry=Series([], dtype: float64), species=[]*)
Bases: object

**geometry**

**get_attribute**(*name, nuclide=None, series=None*)

**get_geometry**()

**get_specie**(*name*)

**get_species**()

**specie**

**species**

## 4.5 fuelsegmentsgroups

Author: Valmor de Almeida dealmeidav@ornl.gov; vfda

Fuel segment

VFdALib support classes

Sat Jun 27 14:46:49 EDT 2015

**class** fuelsegmentsgroups.**FuelSegmentsGroups**(*key=None, fuelSegments=None*)
Bases: object

**AddGroup**(*key, fuelSegments=None*)

**GetAttribute**(*groupKey=None, attributeName=None, nuclideSymbol=None, nuclideSeries=None*)

**GetFuelSegments**(*groupKey=None*)

**HasGroup**(*key*)

**RemoveFuelSegment**(*groupKey, fuelSegment*)

## 4.6 fuelslug

Author: Valmor de Almeida dealmeidav@ornl.gov; vfda

Fuel slug

### 4.6.1 ATTENTION:

This container requires two Phase() containers which are by definition histories. The history is not checked. Therefore any inconsistency will be propagated forward. A fuel slug has two solid phases: cladding and fuel. The user will decide how to best use the underlying history data in the Phase() container of each phase.

VFdALib support classes

Thu Dec 15 16:18:39 EST 2016

**class** fuelslug.**FuelSlug**(*specs=Series([], dtype: float64), fuelPhase= **Phase()**: *quantities*: None *species*: None *history* #time_stamp=1 *history end* @0.0 Series([], Name: 0.0, dtype: float64), claddingPhase= **Phase()**: *quantities*: None *species*: None *history* #time_stamp=1 *history end* @0.0 Series([], Name: 0.0, dtype: float64)*)

Bases: `object`

**GetAttribute**(*name, phase=None, symbol=None, series=None*)

**GetCladdingPhase**()

**GetFuelPhase**()

**GetSpecs**()

**ReduceCladdingVolume**(*dissolvedVolume*)

**ReduceFuelVolume**(*dissolvedVolume*)

**claddingPhase**

**fuelPhase**

**specs**

## 4.7 nuclides

Author: Valmor de Almeida dealmeidav@ornl.gov; vfda

Nuclides container. The purpose of the this container is to store and query a table of nuclides. Typically the table is filled in with data from an ORIGEN calculation or some other fission/transmutation code.

VFdALib support classes

Sat Jun 27 14:46:49 EDT 2015

**class** nuclides.**Nuclides**(*propertyDensities=Empty DataFrame Columns: [] Index: []*)

Bases: `object`

**GetAttribute**(*name, symbol=None, series=None*)

## 4.8 periodictable

Properties of the chemical elements.

Each chemical element is represented as an object instance. Physicochemical and descriptive properties of the elements are stored as instance attributes.

**Author** Christoph Gohlke

**Version** 2015.01.29

Radiochemical data (isotopes) has been added to this table (2015-2016) Origin: http://www.radiochemistry.org/ Valmor F. de Almeida: dealmeidavf@gmail.com; dealmeidav@ornl.gov

### 4.8.1 Requirements

- CPython 2.7 or 3.4

### References


1. http://physics.nist.gov/PhysRefData/Compositions/
2. http://physics.nist.gov/PhysRefData/IonEnergy/tblNew.html
3. http://en.wikipedia.org/wiki/%(element.name)s
4. http://www.miranda.org/~jkominek/elements/elements.db


### Examples

```
>>> from elements import ELEMENTS
>>> len(ELEMENTS)
109
>>> str(ELEMENTS[109])
'Meitnerium'
>>> ele = ELEMENTS['C']
>>> ele.number, ele.symbol, ele.name, ele.eleconfig
(6, 'C', 'Carbon', '[He] 2s2 2p2')
>>> ele.eleconfig_dict
{(1, 's'): 2, (2, 'p'): 2, (2, 's'): 2}
>>> sum(ele.mass for ele in ELEMENTS)
14659.1115599
>>> for ele in ELEMENTS:
...     ele.validate()
...     ele = eval(repr(ele))
```

## 4.9 phase

Phase *history* container. When you think of a phase value, think of that value at a specific point in time.

ATTENTION: The species (list of Specie) AND quantities (list of Quantity) data members have ARBITRARY density values either at an arbitrary point in the history or at no point in the history. This needs to be removed in the future to avoid confusion.

To obtain history values, associated to the phase, at a particular point in time, use the GetValue() method to access the history data frame (pandas) via columns and rows. The corresponding values in species and quantities are OVERRIDEN and NOT to be used through the phase interface.

Author: Valmor F. de Almeida dealmeidav@ornl.gov; vfda Sat Sep 5 01:26:53 EDT 2015

**class** phase.**Phase**(*time_stamp=None*, *species=None*, *quantities=None*)

Bases: object

**AddQuantity**(*newQuant*)

**AddRow**(*try_time_stamp*, *row_values*)

**AddSpecie**(*new_specie*)

**ClearHistory**(*value=0.0*)

**GetActors**()

**GetColumn**(*actor*)

**GetQuantities**()

**GetQuantity**(*name*)

**GetRow**(*try_time_stamp=None*)

**GetSpecie**(*name*)

**GetSpecies**()

**GetTimeStamps**()

**GetValue**(*actor*, *try_time_stamp=None*)

**ResetHistory**(*try_time_stamp=None*, *value=None*)

**ScaleRow**(*try_time_stamp*, *value*)

**SetSpecieId**(*name*, *val*)

**SetValue**(*actor*, *value*, *try_time_stamp=None*)

**WriteHTML**(*fileName*)

**quantities**

**species**

**timeStamps**

## 4.10 quantity

Author: Valmor de Almeida dealmeidav@ornl.gov; vfda

This Quantity class is to be used with other classes in plant-level process modules.

**For unit testing do at the linux command prompt:** python quantity.py

Sat Sep 5 12:51:34 EDT 2015

**class** quantity.**Quantity**(*name='null-quantity'*, *formalName='null-quantity'*, *value=0.0*, *unit='null-unit'*)

Bases: object

**GetFormalName**()

**GetUnit**()

**GetValue**()

**SetFormalName**(*fn*)

**SetName**(*n*)

**SetUnit**(*f*)

**SetValue**(*v*)

**formalName**

**get_name**()

**name**

**unit**

**value**

## 4.11 specie

Author: Valmor de Almeida dealmeidav@ornl.gov; vfda

This Specie class is to be used with other classes in plant-level process modules.

**NB: Species is always used either in singular or plural cases, the class** named here reflects one species. If many species are used in an external context, the species object name can be used without conflict.

**For unit testing do at the linux command prompt:** python specie.py

**NB: The Specie() class encapsulates either the molecular or empirical chemical** formula of a compound. The definition of a chemical species here is extended to ficticious compounds. This is done as follows. Say MAO2 is either a molecular or empirical chemical formula of a ficticious compound denoting minor actinides dioxide. The list of atoms is given as follows:

[‘0.49*Np-237’, ‘0.42*Am-241’, ‘0.08*Am-243’, ‘0.01*Cm-244’, ‘2.0*O-16’]

note the MA forming nuclides add to 1 = 0.49 + 0.42 + 0.08 + 0.01. Therefore the number of atoms in this compound is 3. 1 MA “atom” and 2 O. Note that the total number of “atoms” is obtained by summing all multipliers: 0.49 + 0.42 + 0.08 + 0.01 + 2.0. The nuclide is indicated by the element symbol followed by a dash and the atomic mass number. Here the number of nuclide types is 5 (self._nNuclideTypes).

The numbers preceeding the nuclide symbol before the * will be referred to as multipliers. The sum of the multipliers will add to the number of “atoms” in the formula. WARNING: a multiplier could be in the format 0.00e-00. In this case a hiphen may appear twice, e.g.: 1.549e-09*U-233

Other forms can be used for common true species

[‘Np-237’, ‘2.0*O-16’] or [‘Np-237’, ‘O-16’, ‘O-16’] or [ ‘2*H’, ‘O’ ] or [ ‘H’, ‘O’, ‘H’ ] etc. . .

This code will calculate the molar mass of any species with a given valid atom list using a provided periodic table of chemical elements. The user can also reset the value of the molar mass with a setter method.

Sat May 9 21:40:48 EDT 2015 created; vfda

**class** specie.**Specie**(*name='null', formulaName='null', phase='null', atoms=[], molarCC=0.0, massCC=0.0, flag=None*)

Bases: `object`

**GetAtoms**()

**GetFlag**()

**GetFormula**()

**GetFormulaName**()

**GetMassCC**()

**GetMassCCUnit**()

**GetMolarCC**()

**GetMolarCCUnit**()

**GetMolarGammaPwr**()

**GetMolarGammaPwrUnit**()

**GetMolarHeatPwr**()

**GetMolarHeatPwrUnit**()

**GetMolarMass**()

**GetMolarMassUnit**()

**GetMolarRadioactivity**()

**GetMolarRadioactivityFractions**()

**GetMolarRadioactivityUnit**()

**GetNAtoms**()

**GetNNuclideTypes**()

**GetName**()

**GetPhase**()

**SetAtoms**(*atoms*)

**SetFlag**(*f*)

**SetFormula**(*atoms*)

**SetFormulaName**(*f*)

**SetMassCC**(*v*)

**SetMassCCUnit**(*v*)

**SetMolarCC**(*v*)

**SetMolarCCUnit**(*v*)

**SetMolarGammaPwr**(*v*)

**SetMolarGammaPwrUnit**(*v*)

**SetMolarHeatPwr**(*v*)

**SetMolarHeatPwrUnit**(*v*)

**SetMolarMass**(*v*)

**SetMolarMassUnit**(*v*)

**SetMolarRadioactivity**(*v*)

**SetMolarRadioactivityFractions**(*fracs*)

**SetMolarRadioactivityUnit**(*v*)

**SetName**(*n*)

**SetPhase**(*p*)

**atoms**

**flag**

**formula**

**formulaName**

**massCC**

**massCCUnit**

**molarCC**

**molarCCUnit**

**molarGammaPwr**

**molarGammaPwrUnit**

**molarHeatPwr**

**molarHeatPwrUnit**

**molarMass**

**molarMassUnit**

**molarRadioactivity**

**molarRadioactivityFractions**

**molarRadioactivityUnit**

**nAtoms**

**nNuclideTypes**

**name**

**phase**

## 4.12 stream

Author: Valmor F. de Almeida dealmeidav@ornl.gov; vfda

Stream container

VFdALib support classes

Sat Aug 15 17:24:02 EDT 2015

**class** stream.**Stream**(*timeStamp*, *species=None*, *quantities=None*, *values=0.0*)
Bases: object

**GetActors**()

**GetQuantities**()

**GetQuantity**(*name*)

**GetRow**(*timeStamp=None*)

**GetSpecie**(*name*)

**GetSpecies**()

**GetTimeStamp**()

**GetValue**(*actor*, *timeStamp=None*)

**SetSpecieId**(*name*, *val*)

**SetValue**(*actor*, *value=None*, *timeStamp=None*)

# PYTHON MODULE INDEX

# INDEX

# G

## H



## I



## L



## M

## N



## O



## P



## Q



## R



## S

## T



## U



## V



## W



## X